code differs from the target code, the process comprising:

(a) creating a repository of magic numbers, wherein each magic number is a unique integer representing one and only one variant of a source instruction so that the collection of all the magic numbers represents the set of all valid instructions possible of a source instruction set to form a set of constants associating a unique string with a unique integer, wherein the string shortly describes a variant of the instruction which the magic number is supposed to represent;

(b) creating a repository of templates, wherein each template is a functionally equivalent sequence of target architecture instructions, for each source instruction, each template having temporary native registers for its operations and is compiled at the compile time of the binary dynamic translator as a function and whose name is derived from the corresponding magic number's associated string so that the templates are available as a routine/function at run-time;

(c) creating a table, indexed on magic numbers, containing the references to the corresponding templates so that at run-time the translator can pick up the appropriate template for a given instruction after ascertaining its magic number;

(d) creating a repository of template filler routines in which each template filler routine has the knowledge of both the source and target instructions; causing the template filler routine of a given template to extract dynamic components of the source instruction and fill the extracted items in an in-memory template to replace corresponding place-holders; causing the translator to call the template filler routine for a given instruction at run-time so a template filler routine repository is linked to corresponding templates and magic numbers;

(e) creating a repository of minimal decode/magic-number-routines; causing a minimal decode/magic-number-extractor routines repository to give a raw source instruction and return the magic number of that instruction;

preparing the dynamic binary translator to operate between any processor or platform types by compiling steps (a)-(e) together.

27. (New) The dynamic binary translator system according to claim 26, wherein the dynamic

binary translator maps the source code instructions to the target code instructions using the templates.

28. (New) The dynamic binary translator system according to claim 26, wherein a fill and analysis routine generator arranged to be responsive to the templates for generating fill and analysis routines for identifying the fillable positions in the template by parsing the template and for generating code to extract and deposit fields from the machine instructions in source code into a precompiled template.

29. (New) The dynamic binary translator system according to claim 26, wherein the dynamic binary translator arranged to be responsive to the machine instructions and wherein the dynamic binary translator outputs the translated target code instruction for processing on a target processor using the templates and fill and analysis routines during compilation of the dynamic binary translator.

30. (New) A computer readable medium for performing a build and compilation process of a dynamic binary translator which operates between any processor or platform types and that translates a source code instruction into a target code instruction, having instructions that, when executed by a computer, cause the computer to perform a method comprising:

(a) creating a repository of magic numbers, wherein each magic number is a unique integer representing one and only one variant of a source instruction so that the collection of all the magic numbers represents the set of all valid instructions possible of a source instruction set to form a set of constants associating a unique string with a unique integer, wherein the string shortly describes a variant of the instruction which the magic number is supposed to represent;

(b) creating a repository of templates, wherein each template is a functionally equivalent sequence of target architecture instructions, for each source instruction, each template having temporary native registers for its operations and is compiled at the compile time of the binary dynamic translator as a function and whose name is derived from the corresponding magic number's associated string so that the

templates are available as a routine/function at run-time;

(c) creating a table, indexed on magic numbers, containing the references to the corresponding templates so that at run-time the translator can pick up the appropriate template for a given instruction after ascertaining its magic number;

(d) creating a repository of template filler routines in which each template filler routine has the knowledge of both the source and target instructions; causing the template filler routine of a given template to extract dynamic components of the source instruction and fill the extracted items in an in-memory template to replace corresponding place-holders; causing the translator to call the template filler routine for a given instruction at run-time so a template filler routine repository is linked to corresponding templates and magic numbers;

(e) creating a repository of minimal decode/magic-number-routines; causing a minimal decode/magic-number-extractor routines repository to give a raw source instruction and return the magic number of that instruction;

preparing the dynamic binary translator to operate between any processor or platform types by compiling steps (a)-(e) together.

31. (New) The computer readable medium according to claim 30, wherein the dynamic binary translator maps the source code instructions to the target code instructions using the templates.

32. (New) The computer readable medium according to claim 30, wherein the dynamic binary translator outputs the translated target code instruction for processing on a target processor using the templates and fill and analysis routines during compilation of the dynamic binary translator.